



MICROCONTROLADORES  
PARA APLICACIONES Y  
SISTEMAS BIOMÉDICOS

# INTRODUCCIÓN A LOS WORKFLOWS DE DESARROLLO VCS

Albert Álvarez Carulla

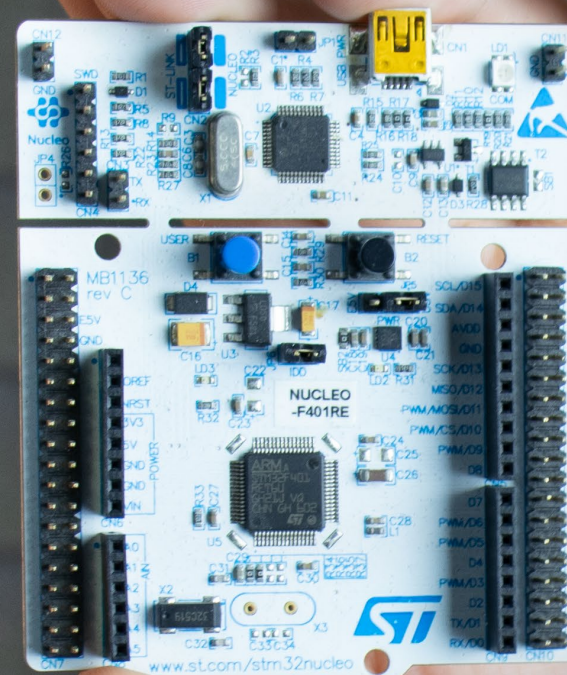
23.03.2021

TheAlbert.dev

@thealbertdev



"Introducción a los workflows de desarrollo: VCS" por Albert Álvarez Carulla se distribuye bajo una [Licencia Creative Commons Atribución-NoComercial-SinDerivadas 4.0 Internacional](#)



# Contenidos

## Introducción a Git

- Problemas resueltos por un VCS
- Tracking de diferencias
- Cómo opera Git
- Conceptos básicos

## Comandos principales

- init
- clone
- add
- status
- commit
- log
- push
- pull
- checkout
- merge

## Workflows

- Master-only flow
- GitHub flow
- Git Flow

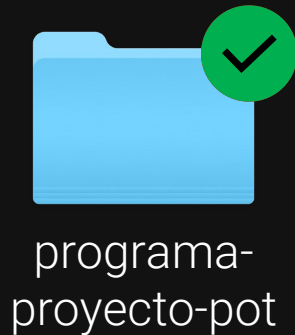
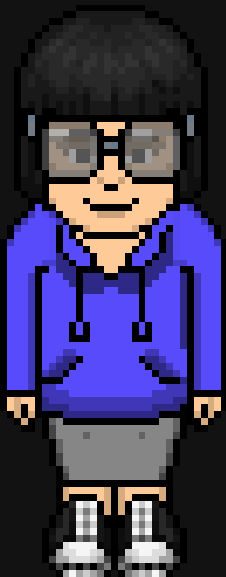
## Introducción a Git

- Problemas resueltos por un VCS
- Tracking de diferencias
- Cómo opera Git
- Conceptos básicos

# Problemas resueltos por un VCS

*"He tocado algo y ahora no funciona"*

**Miriam**



```
cpu_usage.py

#!/usr/bin/env python3

import psutil

if(psutil.cpu_percent(1) > 80):
    print("Uso de CPU excesivo")
else:
    print("Todo OK")
```

# Problemas resueltos por un VCS

*"He tocado algo y ahora no funciona"*

**Miriam**



```
cpu_usage.py

#!/usr/bin/env python3

import psutil

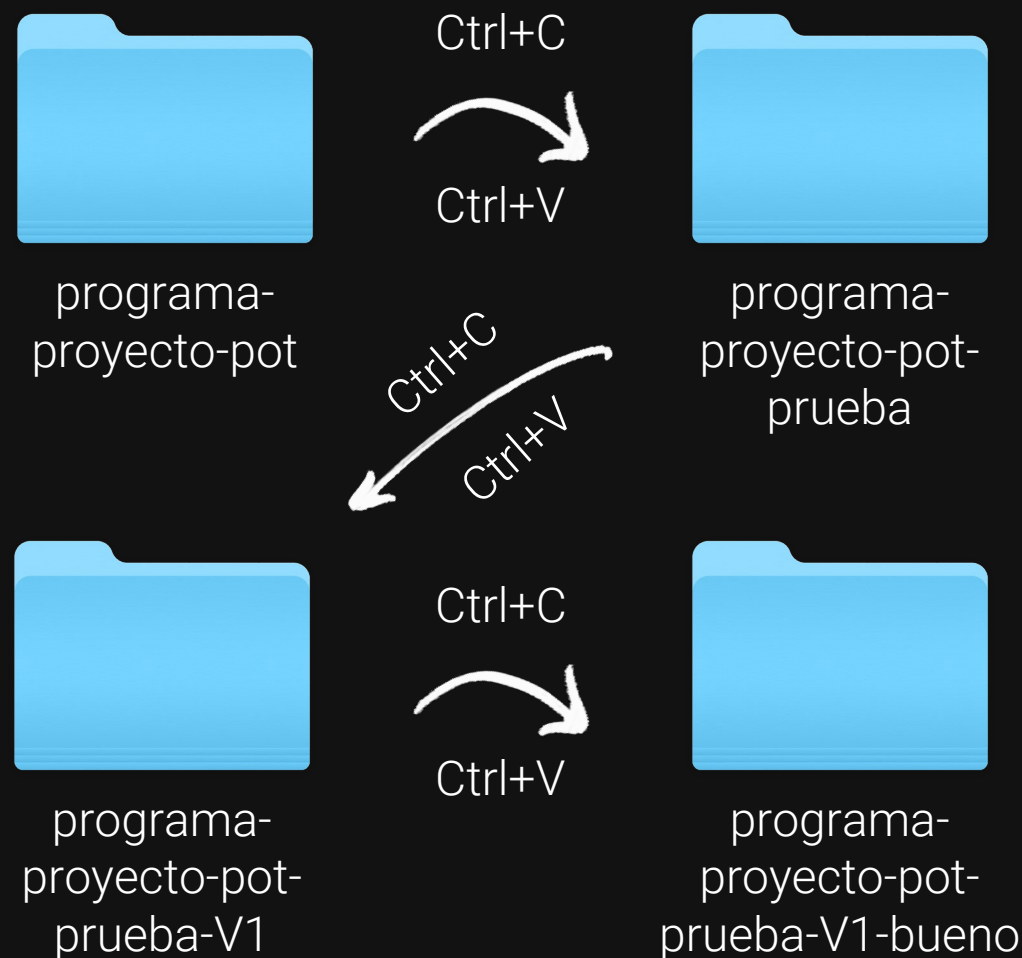
if(psutil.cpu_percent() > 80):
    print("Uso de CPU excesivo")
else:
    print("Todo OK")
```

# Problemas resueltos por un VCS

*"Eeeh... ¿Cuál era el bueno?"*

*"Me he quedado sin espacio. ¿Qué elimino?"*

**Olga**



# Problemas resueltos por un VCS

*“Eeeh... ¿Cuál era el bueno?”*

*“Me he quedado sin espacio. ¿Qué elimino?”*

Olga



# 2.32 GB



programa-  
proyecto-pot



programa-  
proyecto-pot-  
prueba



programa-  
proyecto-pot-  
prueba-V1



programa-  
proyecto-pot-  
prueba-V1-  
bueno



programa-  
proyecto-pot-  
prueba-V1-  
bueno-  
definitivo



prog



programa-  
proyecto-pot-  
borrar



programa-  
proyecto-pot-  
código-  
stackoverflow



programa-  
proyecto-pot-  
no-build



cacacaca



no-utilizar



prog  
proye  
fina



programa-  
proyecto-pot-  
principal



programa-  
proyecto-pot-  
refactored



programa-  
proyecto-pot-  
ahora-si



programa-  
proyecto-pot-  
definitivo1



programa-  
proyecto-pot-  
definitivo2

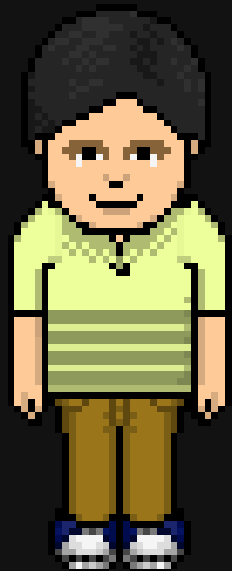


prog  
proye  
defi  
def

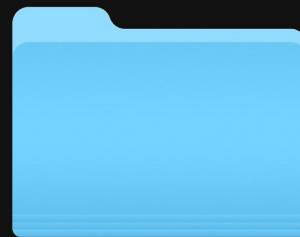
# Problemas resueltos por un VCS

*"Fran no hace ni el huevo."*

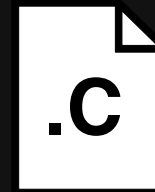
*"Es que hasta que no acabes no puedo poner mi parte."*



**Fran**



programa-  
proyecto-pot

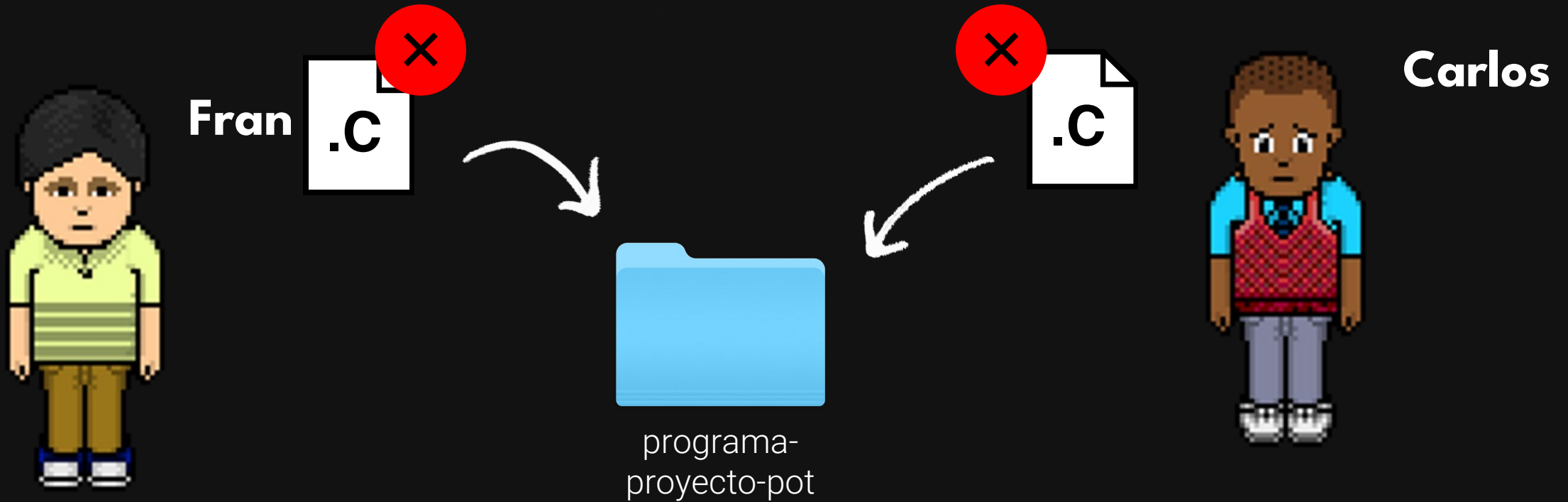


**Carlos**



# Problemas resueltos por un VCS

*"Al unirlo todo, la hemos liado..."*



# Tracking de diferencias

**Miriam**



```
cpu_usage.py

#!/usr/bin/env python3

import psutil

if(psutil.cpu_percent() > 80):
    print("Uso de CPU excesivo")
else:
    print("Todo OK")
```

# Tracking de diferencias

Miriam



Versión anterior

```
cpu_usage_old.py

#!/usr/bin/env python3

import psutil

if(psutil.cpu_percent(1) > 80):
    print("Uso de CPU excesivo")
else:
    print("Todo OK")
```

Versión actual

```
cpu_usage.py

#!/usr/bin/env python3

import psutil

if(psutil.cpu_percent() > 80):
    print("Uso de CPU excesivo")
else:
    print("Todo OK")
```

# Tracking de diferencias

**Miriam**



```
Terminal

$ diff cpu_usage_old.py cpu_usage.py
```

# Tracking de diferencias

**Miriam**



```
Terminal

$ diff cpu_usage_old.py cpu_usage.py
5c5
< if(psutil.cpu_percent(1) > 80):
---
> if(psutil.cpu_percent() > 80):
```

# Tracking de diferencias

Solucionamos el problema de poder solucionar incorporación de bugs en nuestro proyecto

**Miriam**



```
Terminal

$ diff -u cpu_usage_old.py cpu_usage.py
--- cpu_usage_old.py      2021-03-22
13:21:27.756645200 +0100
+++ cpu_usage.py          2021-03-22
13:21:35.750272100 +0100
@@ -2,7 +2,7 @@

import psutil

-if(psutil.cpu_percent(1) > 80):
+if(psutil.cpu_percent() > 80):
    print("Uso de CPU excesivo")
else:
    print("Todo OK")
\ No newline at end of file
```

# Tracking de diferencias

# 2.32 GB

Olga



programa-  
proyecto-pot



programa-  
proyecto-pot-  
prueba



programa-  
proyecto-pot-  
prueba-V1



programa-  
proyecto-pot-  
prueba-V1-  
bueno



programa-  
proyecto-pot-  
prueba-V1-  
bueno-  
definitivo



prog



programa-  
proyecto-pot-  
borrar



programa-  
proyecto-pot-  
código-  
stackoverflow



programa-  
proyecto-pot-  
no-build



cacacaca



no-utilizar



prog  
proye  
fina



programa-  
proyecto-pot-  
principal



programa-  
proyecto-pot-  
refactored



programa-  
proyecto-pot-  
ahora-si



programa-  
proyecto-pot-  
definitivo1



programa-  
proyecto-pot-  
definitivo2



prog  
proye  
defi  
def

# Tracking de diferencias

Olga



```
Terminal

$ diff cpu_usage_old.py cpu_usage.py >
cpu_usage_sv5310.diff
```



# Tracking de diferencias

Olga



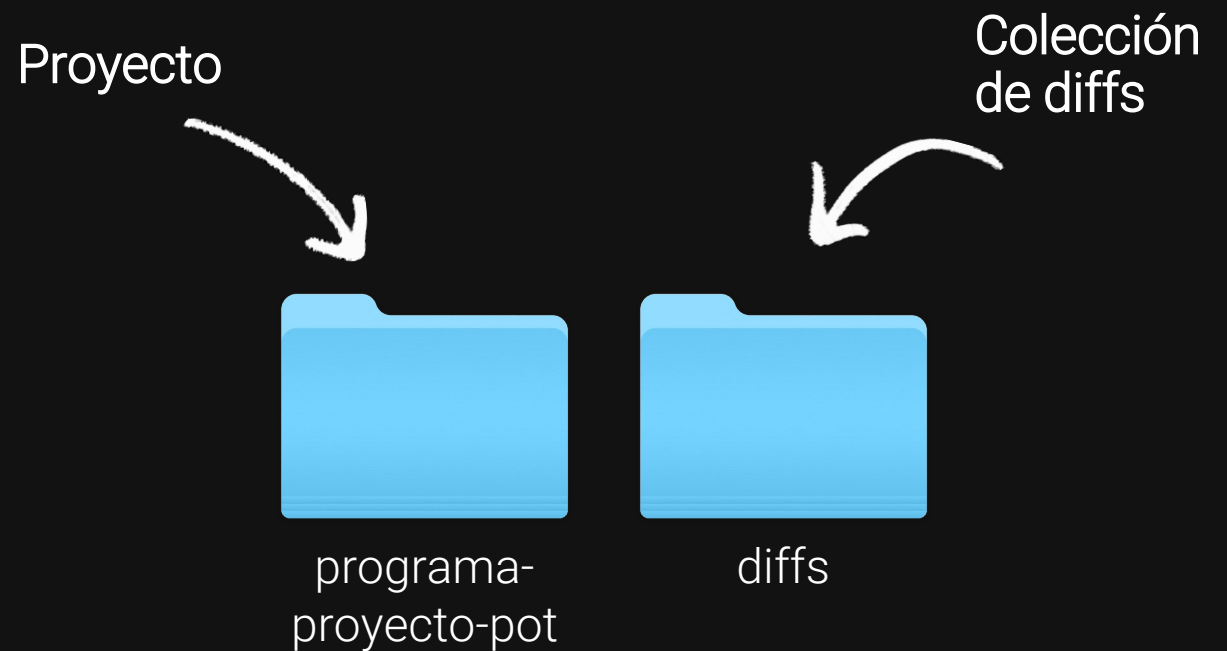
```
cpu_usage_sv5310.diff

5c5
< if(psutil.cpu_percent(1) > 80):
---
> if(psutil.cpu_percent() > 80):
```

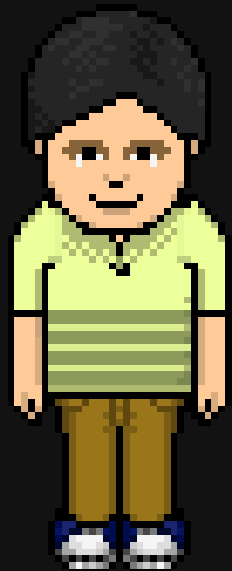
# Tracking de diferencias

Tenemos un solo proyecto con los cambios monitorizados  
Además, reducimos el peso del proyecto

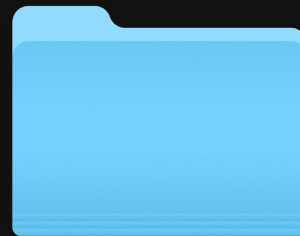
**Olga**



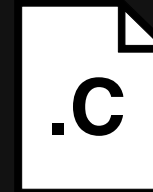
# Tracking de diferencias



**Fran**



programa-  
proyecto-pot



**Carlos**

# Tracking de diferencias



```
cpu_usage.py

#!/usr/bin/env python3

import psutil
```

# Tracking de diferencias



```
cpu_usage.py

#!/usr/bin/env python3

import psutil

if(psutil.cpu_percent(1) > 80):
    print("Uso de CPU excesivo")
else:
    print("Todo OK")
```

# Tracking de diferencias



**Carlos**

```
Terminal (Fran)  
  
$ diff cpu_usage.py cpu_usage_fran.py >  
cpu_usage_fran.patch
```



**Fran**



# Tracking de diferencias



**Carlos**

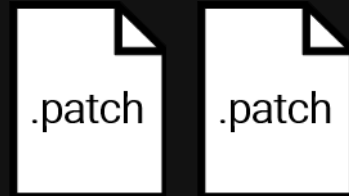
```
Terminal (Fran)  
  
$ diff cpu_usage.py cpu_usage_fran.py >  
cpu_usage_fran.patch
```

```
Terminal (Carlos)  
  
$ diff cpu_usage.py cpu_usage_fran.py >  
cpu_usage_fran.patch
```



**Fran**

# Tracking de diferencias

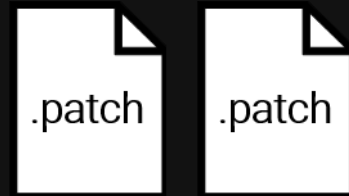


```
Terminal (Carlos)

$ patch cpu_usage.py < cpu_usage_fran.patch
$ patch cpu_usage.py < cpu_usage_carlos.patch
```



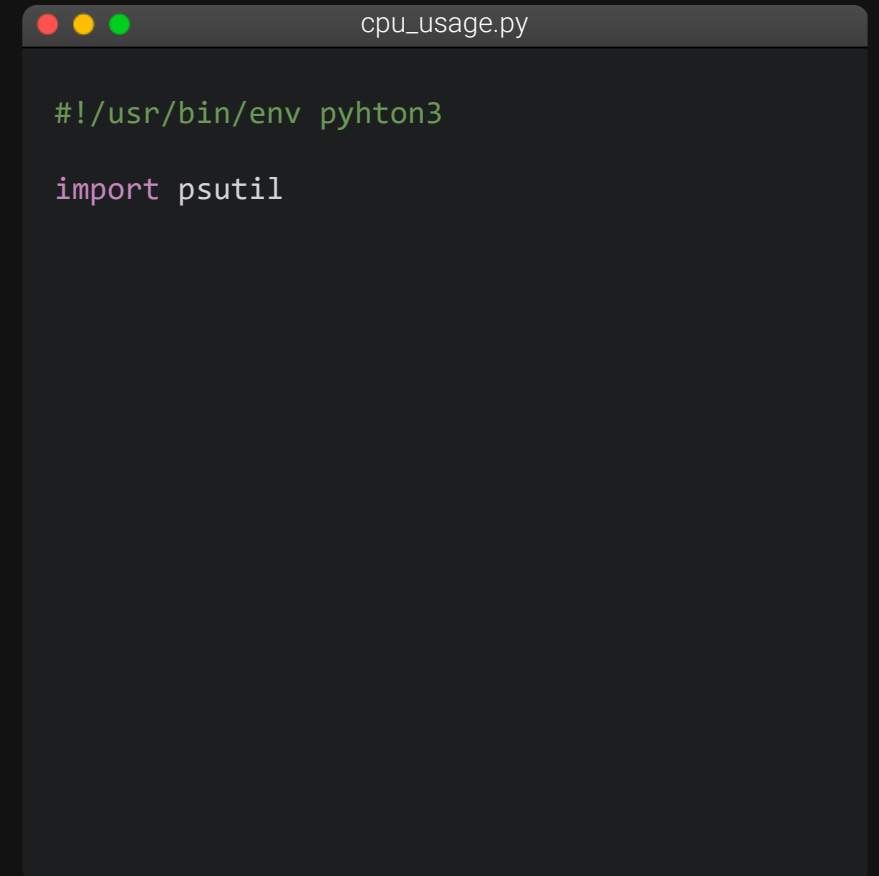
# Tracking de diferencias



```
Terminal (Carlos)

$ patch cpu_usage.py < cpu_usage_fran.patch
$ patch cpu_usage.py < cpu_usage_carlos.patch
```

# Tracking de diferencias



```
#!/usr/bin/env python3
import psutil
```

A screenshot of a code editor window titled "cpu\_usage.py". The window has a dark gray background and a title bar with three colored window control buttons (red, yellow, green). The code is written in a light green monospace font. The first line is a shebang line: `#!/usr/bin/env python3`. The second line is an import statement: `import psutil`.

# Tracking de diferencias

Podemos trabajar simultáneamente en un mismo proyecto



**Carlos**

```
cpu_usage.py

#!/usr/bin/env python3

import psutil

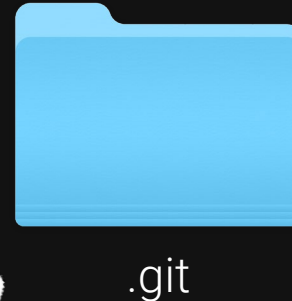
if(psutil.cpu_percent(1) > 80):
    print("Uso de CPU excesivo")
else:
    print("Todo OK")
```

# Version Control System

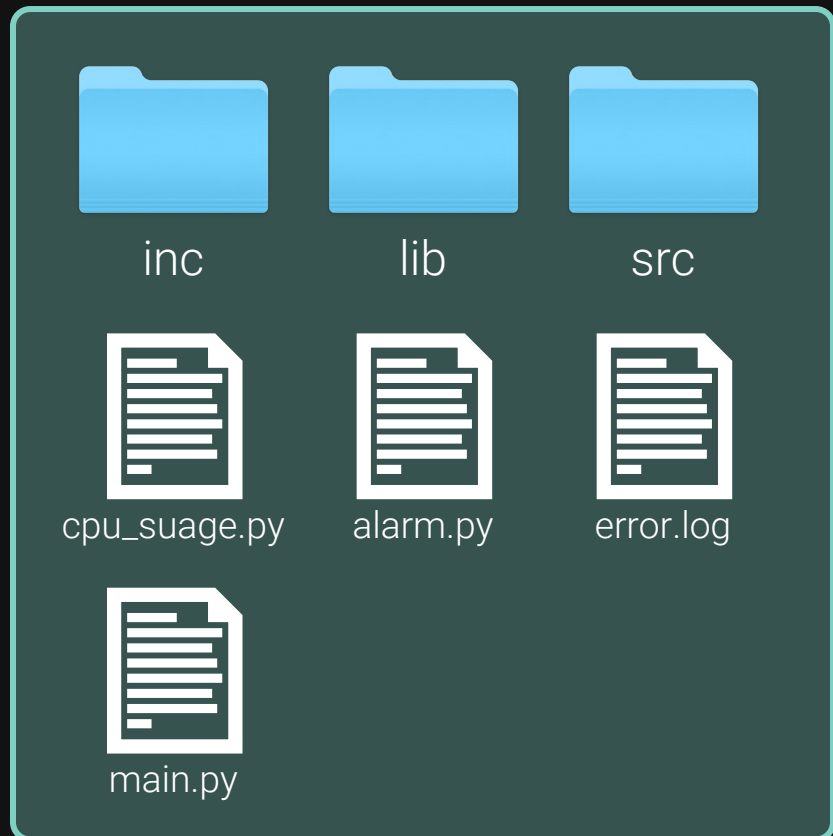


# Cómo opera git y conceptos básicos

“Colección de diffs”



## working tree



# Jimmy



971bdaac18682968ff244bab696a90c945a296a5



971bda

# commit



# Jimmy



971bda



0a4db2



4ddbd1



ffc5c0



# Jimmy



971bda



0a4db2



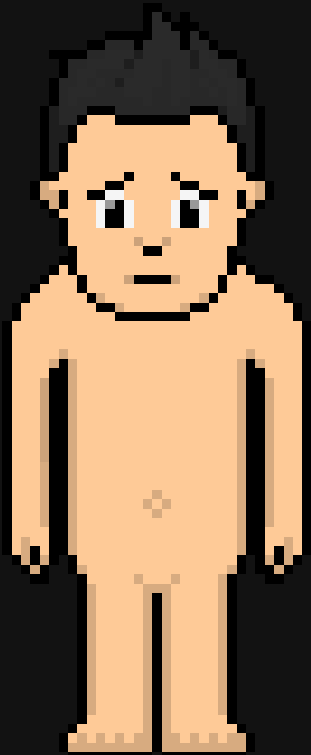
4ddbd1



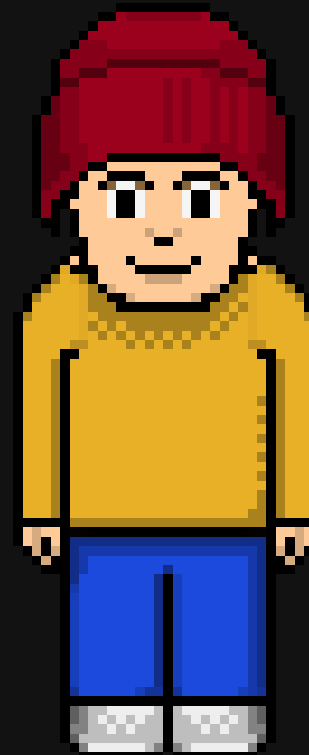
ffc5c0



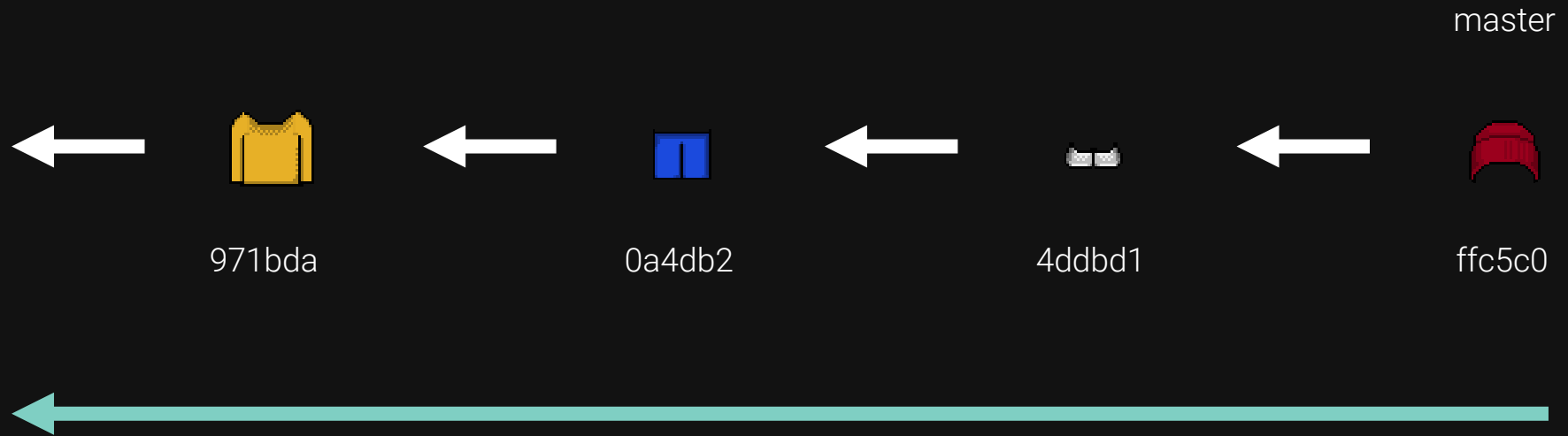
# Jimmy



=



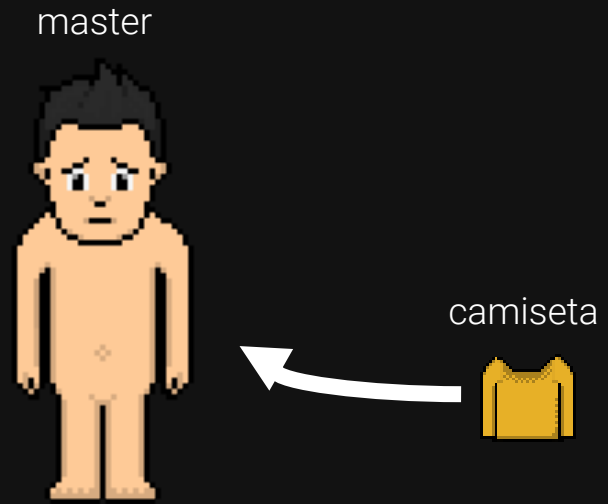
# Jimmy

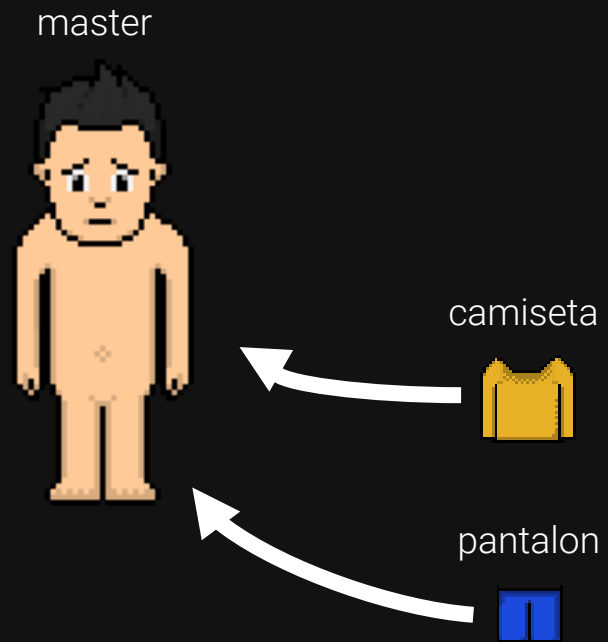


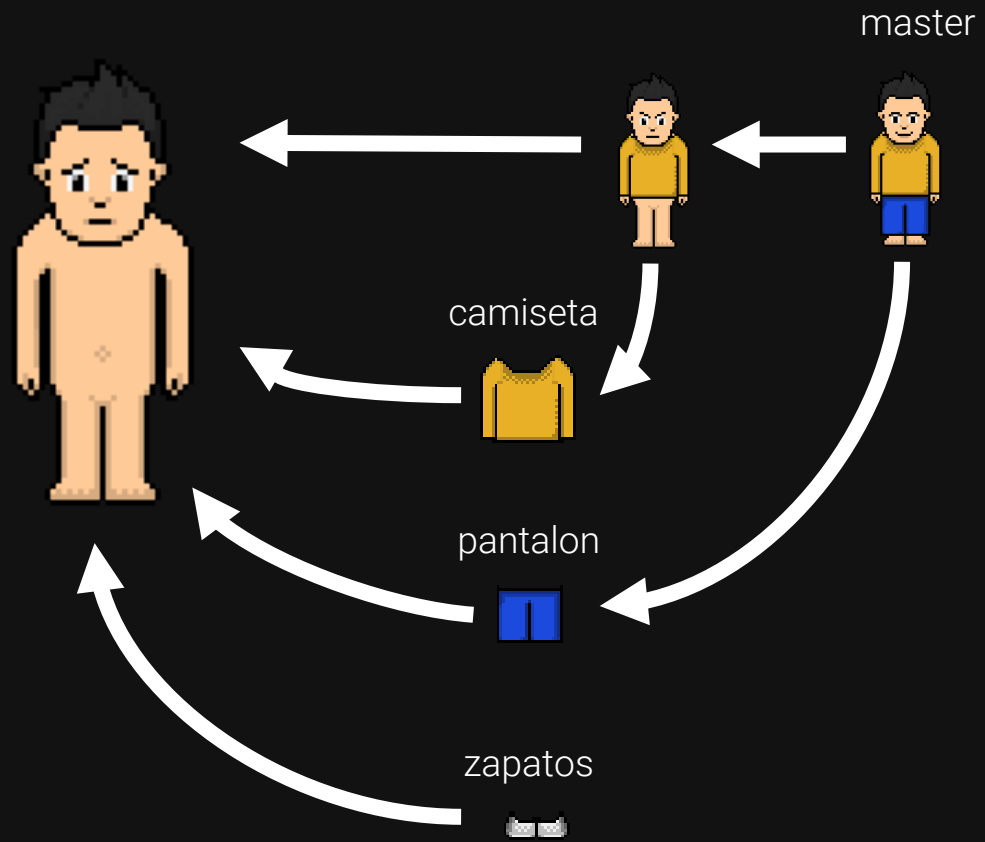
# branch

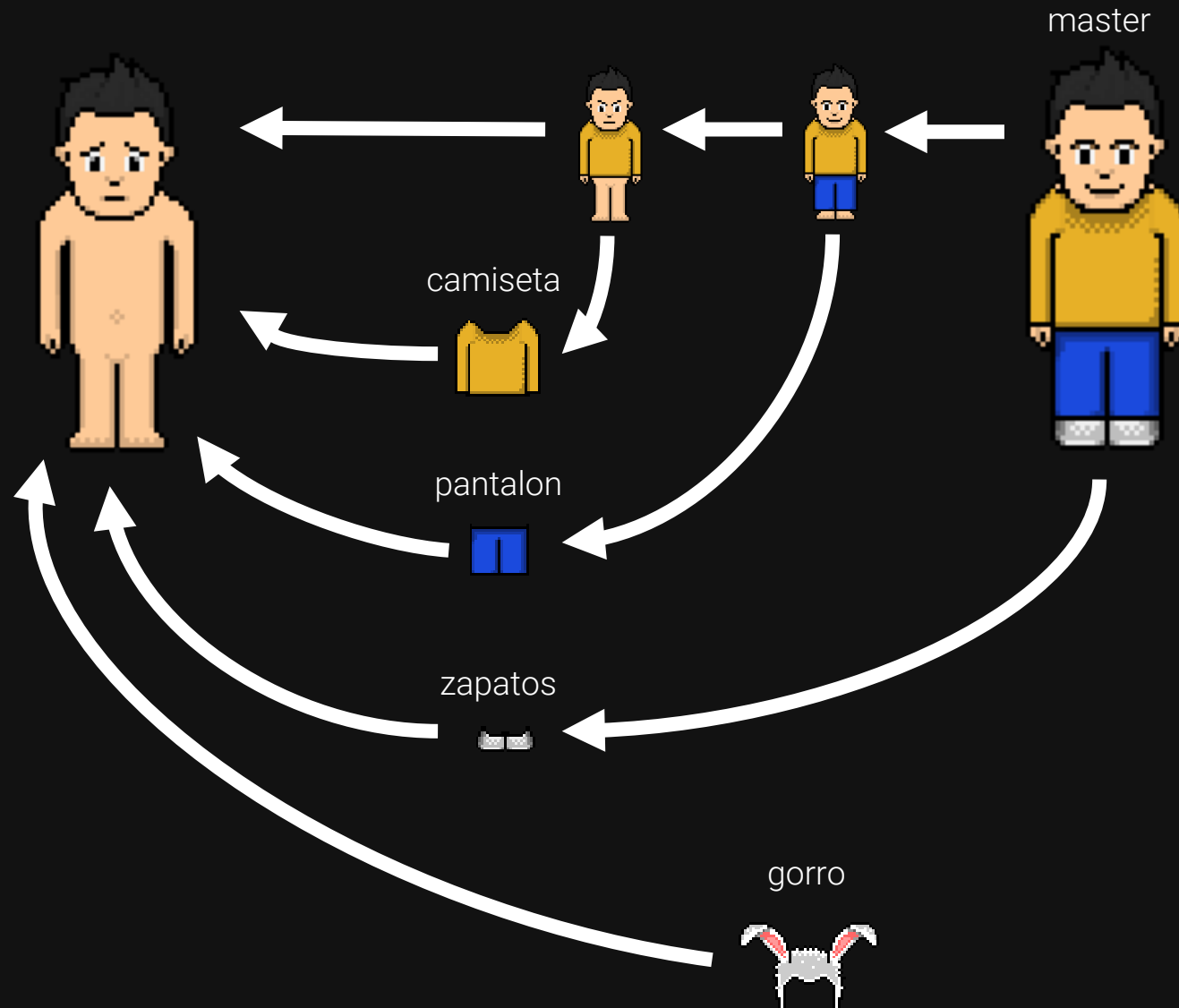
master



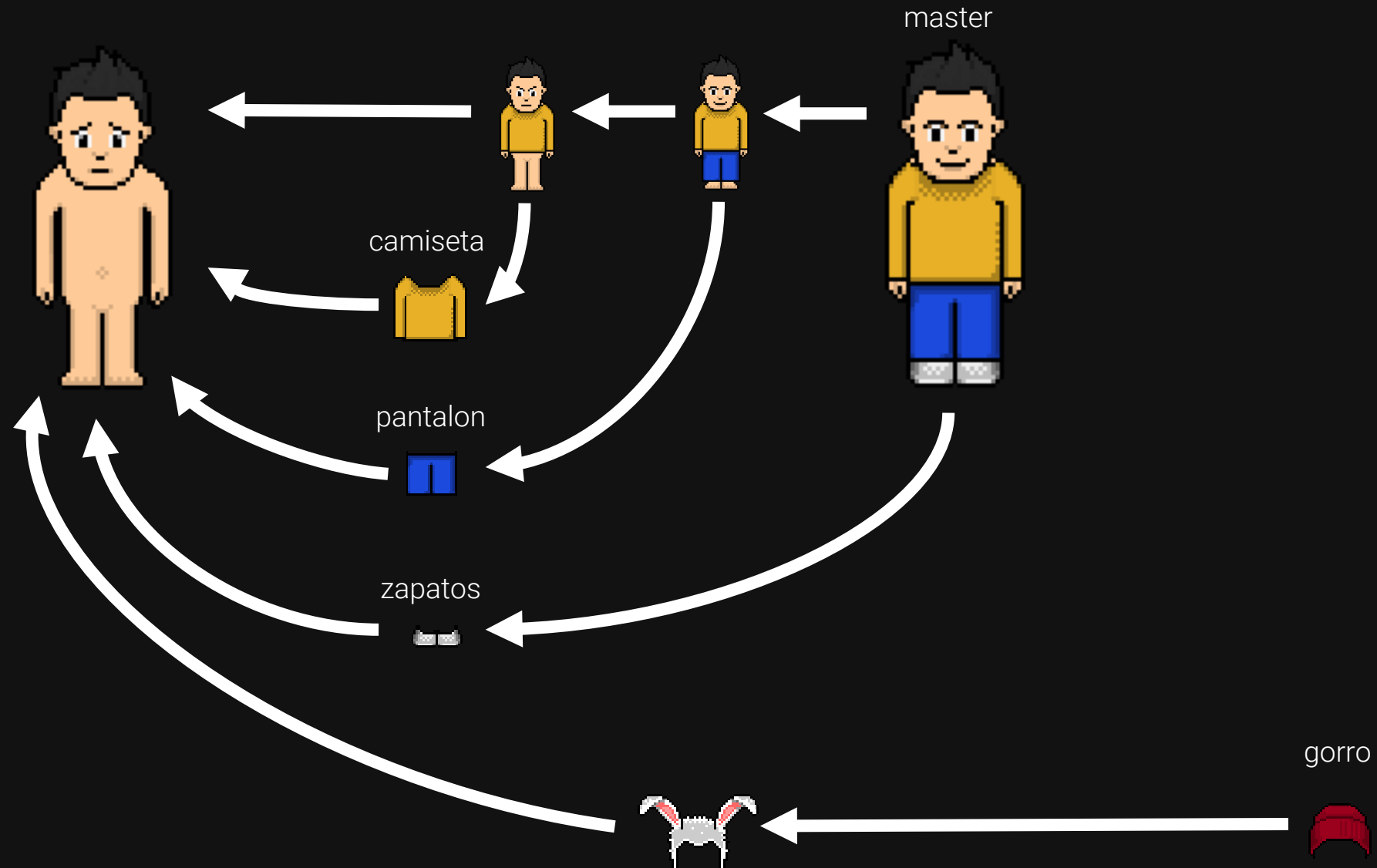




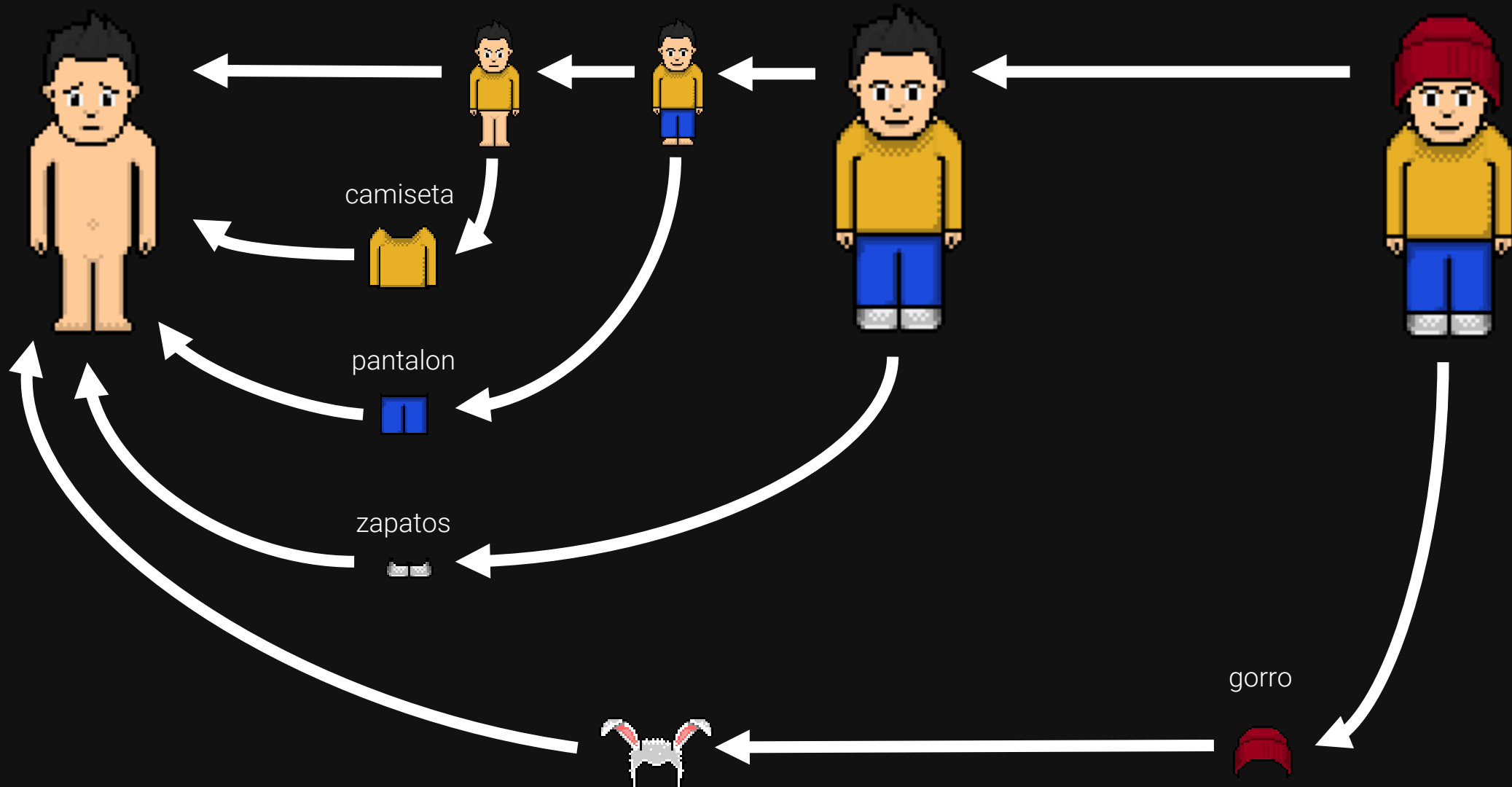




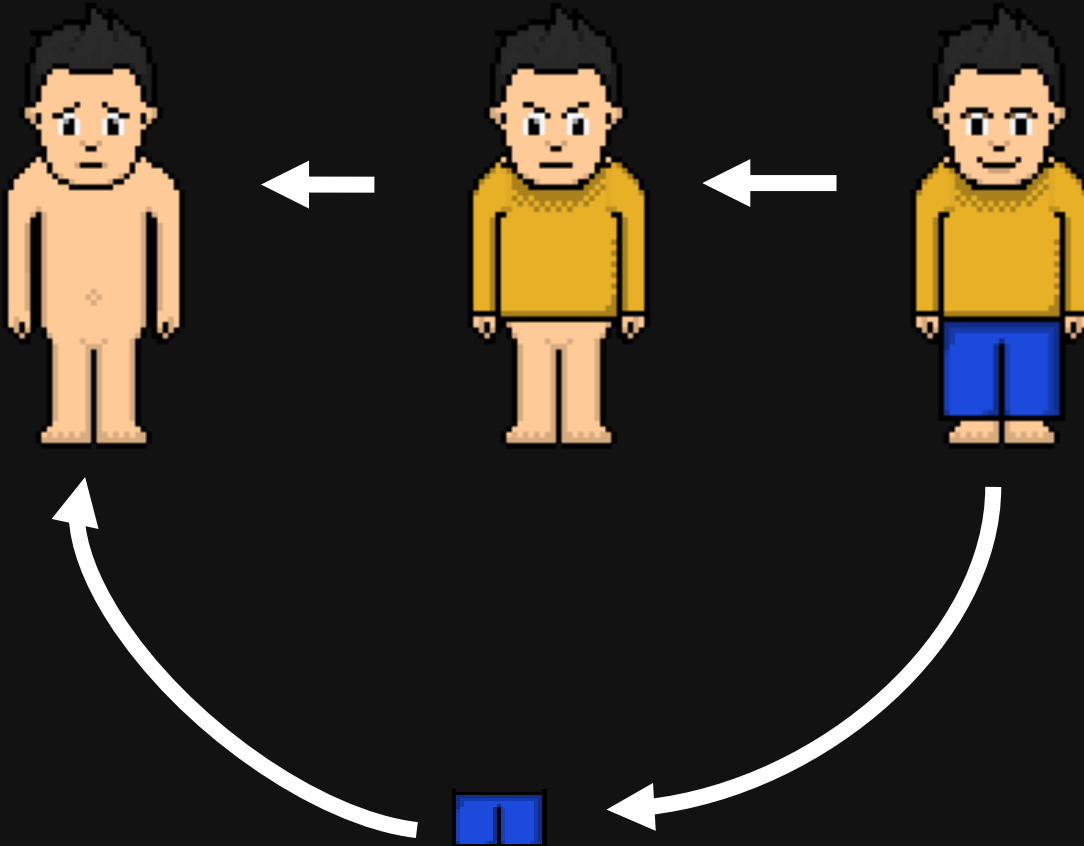




master



# Jimmy



# merge



## Comandos principales

- init
- clone
- add
- status
- commit
- log
- push
- pull
- checkout
- merge

# git init

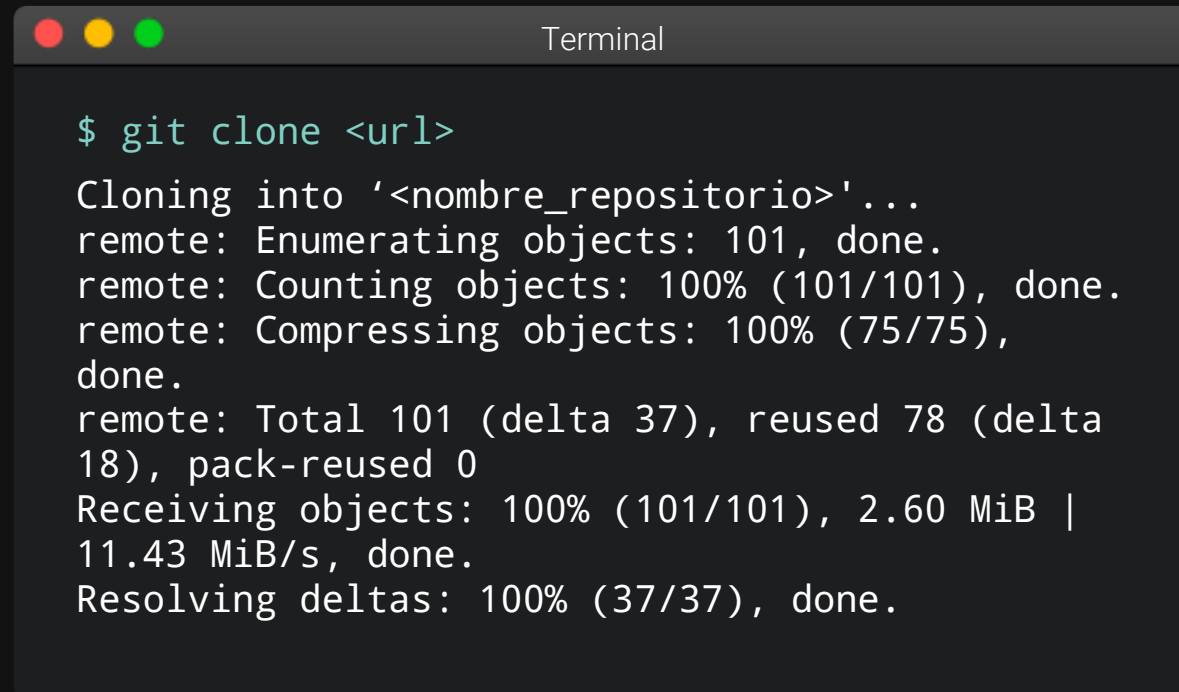
Inicializa/crea un repositorio git.

A terminal window with a dark background and a title bar labeled "Terminal". The title bar has three colored window control buttons (red, yellow, green) on the left. The terminal content shows the command "\$ git init" in green text, followed by the output "Initialized empty Git repository in \*" in white text.

```
$ git init
Initialized empty Git repository in *
```

# git clone <url>


Descarga un repositorio git en un servidor remoto. Por defecto, el nombre de ese remoto es 'origin'.

A terminal window with a dark background and light gray text. The title bar shows three colored window control buttons (red, yellow, green) and the word "Terminal". The command prompt shows the execution of 'git clone <url>'. The output shows the cloning process: enumerating, counting, and compressing objects, followed by receiving objects and resolving deltas, all completed successfully.

```
$ git clone <url>
Cloning into '<nombre_repositorio>'...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (75/75),
done.
remote: Total 101 (delta 37), reused 78 (delta
18), pack-reused 0
Receiving objects: 100% (101/101), 2.60 MiB |
11.43 MiB/s, done.
Resolving deltas: 100% (37/37), done.
```

# git add <file>

Añade uno o más archivos al área de stage para, posteriormente, poder registrar sus cambios.

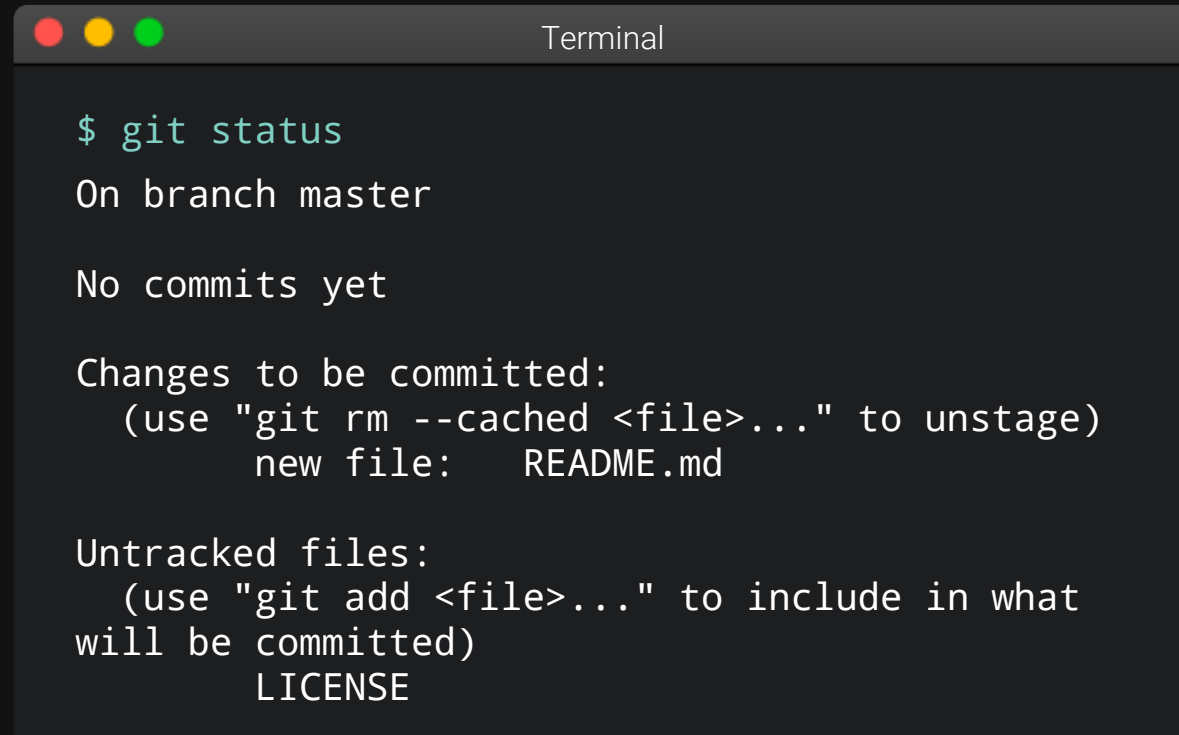
A terminal window with a dark gray background and a title bar labeled "Terminal". The title bar has three colored window control buttons (red, yellow, green) on the left. The terminal content shows a prompt character "\$" followed by the command "git add <file>" in a light blue monospace font.

```
$ git add <file>
```



# git status

Indica los archivos que han sido modificados y cuyos cambios se han incorporado en el área de stage o no.

A terminal window with a dark background and a title bar labeled "Terminal". The window contains the output of the "git status" command. The text is as follows:

```
$ git status
On branch master

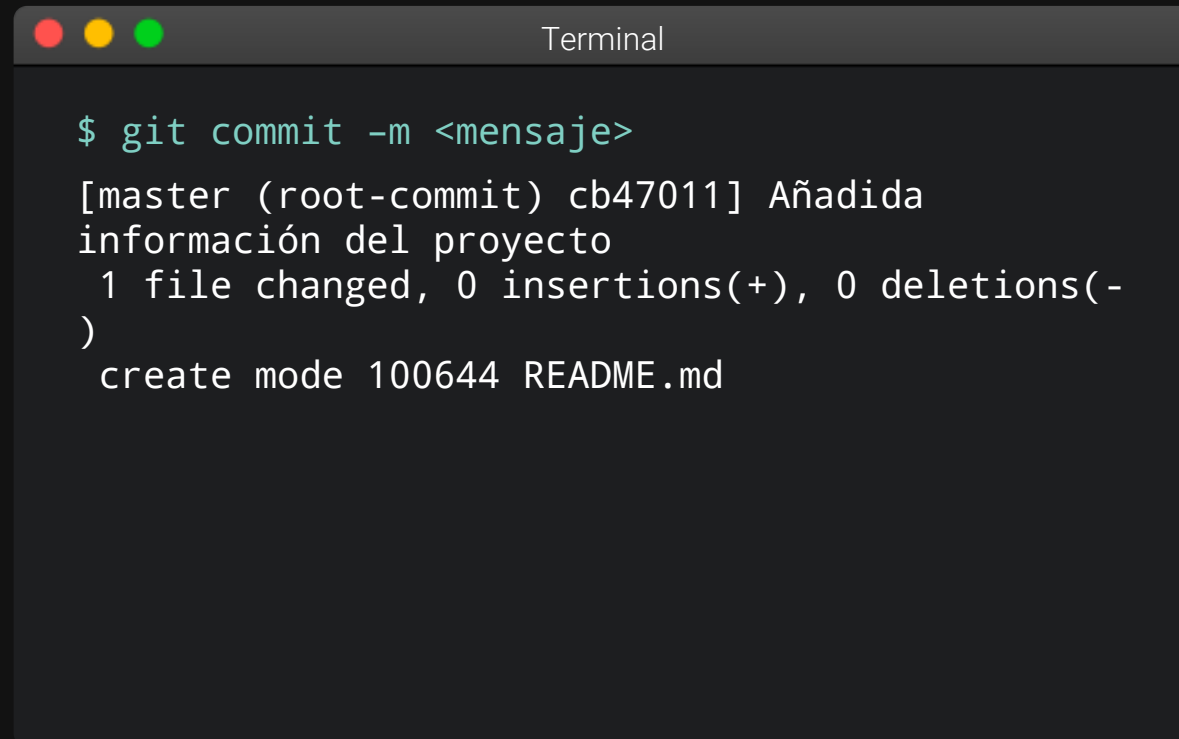
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md

Untracked files:
  (use "git add <file>..." to include in what
will be committed)
    LICENSE
```

# git commit -m <mensaje>

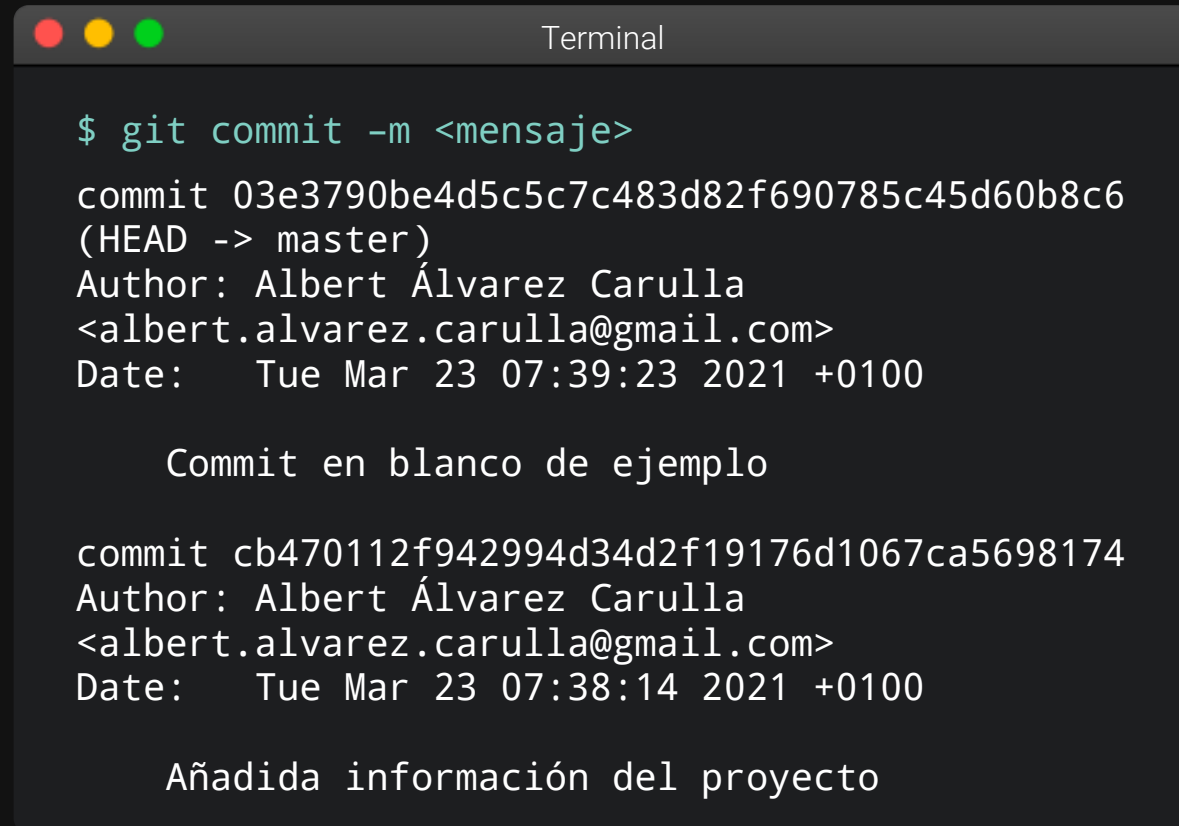
Realiza un commit en la rama actual con los cambios añadidos en el área de stage. Se utiliza '-m' para indicar, posteriormente, el mensaje/descripción asignado a ese commit. Si no se indica el mensaje, se abre el editor de texto configurado para añadirlo.

A terminal window with a dark background and a title bar labeled "Terminal". The window contains the following text:

```
$ git commit -m <mensaje>
[master (root-commit) cb47011] Añadida
información del proyecto
 1 file changed, 0 insertions(+), 0 deletions(-)
)
create mode 100644 README.md
```

# git log

Muestra el histórico del repositorio.



```
$ git commit -m <mensaje>

commit 03e3790be4d5c5c7c483d82f690785c45d60b8c6
(HEAD -> master)
Author: Albert Álvarez Carulla
<albert.alvarez.carulla@gmail.com>
Date:   Tue Mar 23 07:39:23 2021 +0100

    Commit en blanco de ejemplo

commit cb470112f942994d34d2f19176d1067ca5698174
Author: Albert Álvarez Carulla
<albert.alvarez.carulla@gmail.com>
Date:   Tue Mar 23 07:38:14 2021 +0100

    Añadida información del proyecto
```

# git push

Sube los cambios de nuestro repositorio local al remoto o origin.

```
Terminal

$ git push

Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 344 bytes | 344.00
KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-
reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Albert-Alvarez/demo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch
'master' from 'origin'.
```

# git pull

Descarga los cambios del repositorio remoto o origin al local.

```
Terminal

$ git pull

remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0),
pack-reused 0
Unpacking objects: 100% (3/3), 628 bytes | 1024
bytes/s, done.
From https://github.com/Albert-Alvarez/demo
   03e3790..2a62f26  master    ->
origin/master
Updating 03e3790..2a62f26
Fast-forward
 README.md | 1 +
1 file changed, 1 insertion(+)
```

# git checkout <branch | commit\_id>

Salta entre las diferentes versiones del repositorio modificando los contenidos de nuestro working tree. Se debe de indicar o el nombre de la rama o el commit id (6 primeros dígitos). Podemos crear una rama nueva con el parámetro '-b'.

Ver también 'git switch'.

```
Terminal

$ git checkout 03e379
Note: switching to '03e379'.

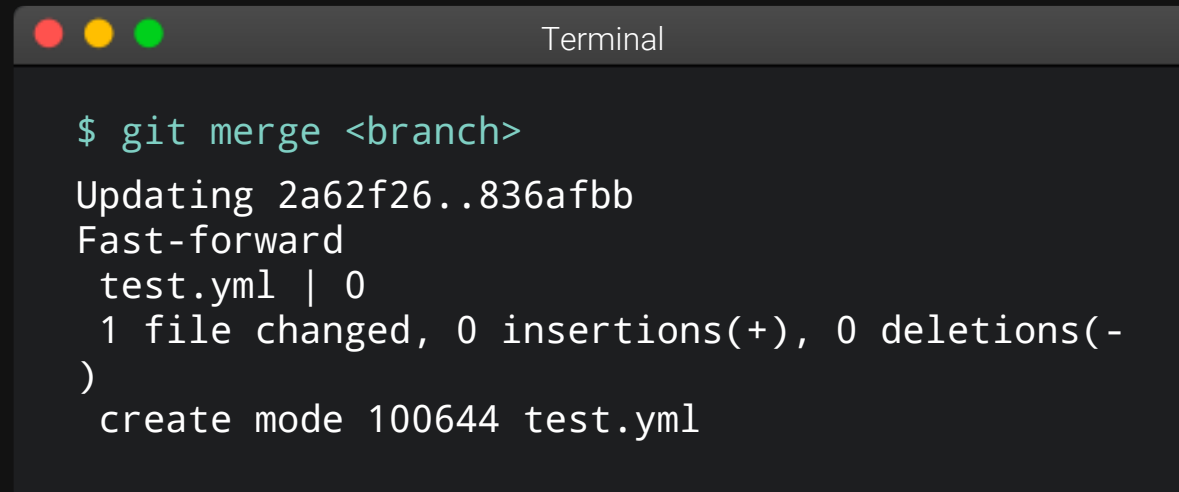
You are in 'detached HEAD' state. You can look around, make
experimental
changes and commit them, and you can discard any commits
you make in this
state without impacting any branches by switching back to a
branch.

If you want to create a new branch to retain commits you
create, you may
do so (now or later) by using -c with the switch command.
Example:

    git switch -c <new-branch-name>
...
```

# git merge <branch>

Fusión de dos ramas. Incorpora a nuestra rama actual los cambios de la rama que se indica en el comando.

A terminal window with a dark background and a title bar labeled "Terminal". The window contains the output of the command "git merge <branch>". The output shows a fast-forward merge of branch 2a62f26 into the current branch 836afbb, resulting in one file change (test.yml) with no insertions or deletions.

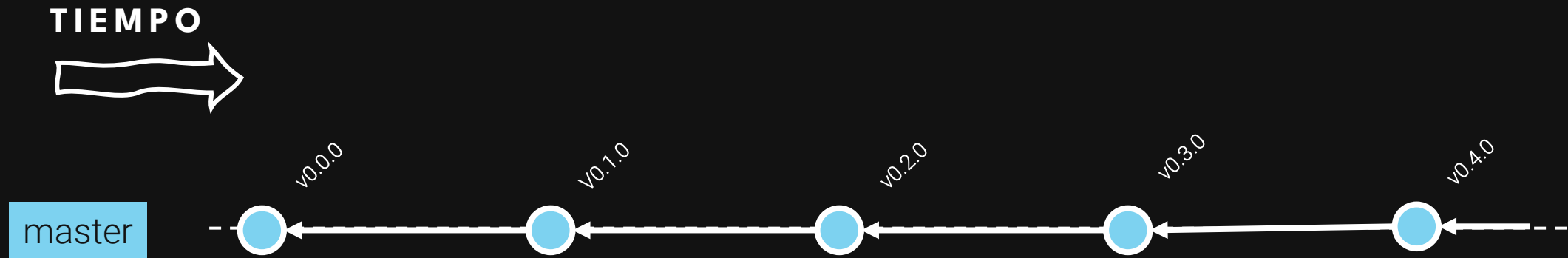
```
$ git merge <branch>
Updating 2a62f26..836afbb
Fast-forward
 test.yml | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.yml
```

## Workflows

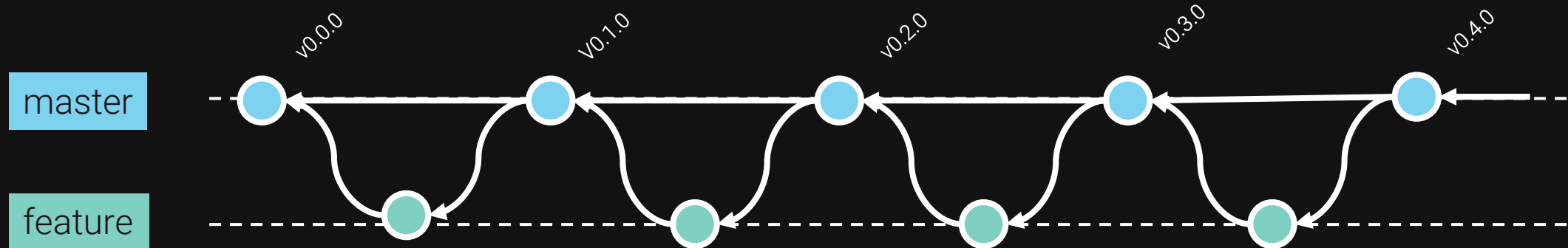
- Master-only flow
- GitHub flow
- Git Flow



# Master-only flow

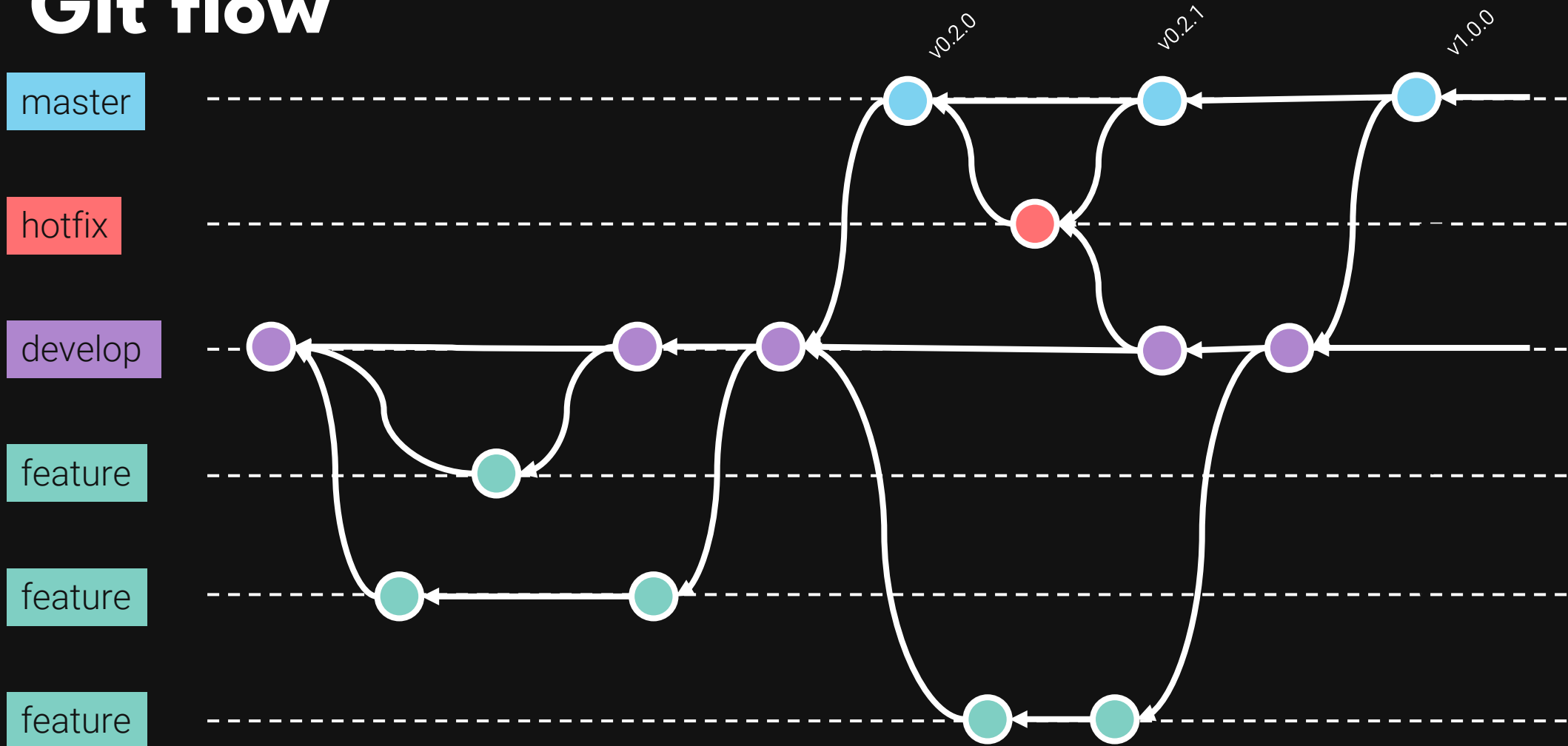


# GitHub flow



[Understanding the GitHub flow · GitHub Guides](#)

# Git flow



[A successful Git branching model » nvie.com](https://nvie.com/articles/a-successful-git-branching-model)



# Introducción a los workflows de desarrollo (VCS)

Albert Álvarez Carulla

23.03.2021

TheAlbert.dev



@thealbertdev



"Introducción a los workflows de desarrollo (VCS)" por Albert Álvarez Carulla se distribuye bajo una [Licencia Creative Commons Atribución-NoComercial-SinDerivadas 4.0 Internacional](#)